BodyCloud: a SaaS approach for community body sensor networks

Elsevier Journal on Future Generation Computer Systems, Volume 35, June 2014, pp.62–79 [in press]

BodyCloud: A SaaS Approach for Community Body Sensor Networks

Giancarlo Fortino, Daniele Parisi, Vincenzo Pirrone
DIMES – University of Calabria
87036 Rende (CS), Italy
g.fortino@unical.it

Giuseppe Di Fatta
SSE – The University of Reading
Reading, RG6 6AX, UK
G.DiFatta@reading.ac.uk

***Abstract*—Body Sensor Networks (BSNs) have been recently introduced for the remote monitoring of human activities in a broad range of application domains, such as health care, emergency management, fitness and behaviour surveillance. BSNs can be deployed in a community of people and can generate large amounts of contextual data that require a scalable approach for storage, processing and analysis. Cloud computing can provide a flexible storage and processing infrastructure to perform both online and offline analysis of data streams generated in BSNs. This paper proposes BodyCloud, a SaaS approach for community BSNs that supports the development and deployment of Cloud-assisted BSN applications. BodyCloud is a multi-tier application-level architecture that integrates a Cloud computing platform and BSN data streams middleware. BodyCloud provides programming abstractions that allow the rapid development of community BSN applications. This work describes the general architecture of the proposed approach and presents a case study for the real-time monitoring and analysis of cardiac data streams of many individuals.**

Keywords: Body Sensor Networks; Cloud Computing; Software Engineering; SaaS; Sensor Data as a Service; Analytics as a Service

1. Introduction

Wireless Sensor Networks (WSNs) [4, 58] consist of spatially distributed, interconnected, wireless sensor nodes, which are typically employed to cooperatively monitor physical, environmental, or human conditions such as temperature, sound, vibration, pressure, motion, heart rate and blood pressure. When WSNs are deployed in large scale applications, they can generate a large amount of dense, in-situ contextual data. The capabilities of WSNs are not just limited to observing and forwarding raw sensor readings, they enable innovative real-time applications in a wide range of domains. They can support different applications and services, ranging from home automation to process monitoring, healthcare analysis, weather forecasting, military logistics, and traffic control.

Body Sensor Networks (BSNs) [16, 57] are a particular type of WSNs that have emerged as a result of the advancements of ubiquitous and increasingly powerful wearable devices. BSNs provide a platform

for many human-centered applications, spanning from health care to sports performance monitoring, gaming and social networking. There is currently an enormous public interest in products that allow individuals, ranging from children to elders, to monitor and improve their health and lifestyle. In a common healthcare scenario, assisted livings are monitored by BSNs to gather data streams to process them in real-time [23] and to store them in remote medical data repositories for offline analysis. This scenario implies a huge amount of data being transmitted, stored and analyzed.

In the coming years, BSNs are likely to be exploited to enable implicit social interactions among people who exchange private/public information through their worn BSN nodes when they come into contact [9]. BSNs of co-located people can also be utilized as an infrastructure of mobile sensors to support other context-aware applications such as disaster management, medical emergencies and mass event monitoring. In such contexts, the management and the cooperation [8] of a large number of BSNs is an important aspect still to be addressed.

The huge amount of data that is expected to be generated by BSNs requires a powerful and scalable storage and processing infrastructure that is able to support both online and offline analysis of data streams. Such requirements can be met by an integrated platform based on Cloud computing [24] with the following characteristics: (a) the ability to utilize heterogeneous sensors; (b) scalability of data storage; (c) scalability of processing power for different kinds of analysis; (d) global access to the processing and storage infrastructure; (e) easy sharing of results; and (f) pay-as-you-go pricing for using BSN services.

In this paper, we propose a system architecture, BodyCloud, that integrates BSN services with a Cloud computing infrastructure. BodyCloud is a SaaS architecture that supports the storage and management of sensor data streams and the processing (online and offline analysis) of the stored data using software services hosted in the Cloud. In particular, BodyCloud endeavors to support several cross-disciplinary applications and specialized processing tasks. It enables large-scale data sharing and collaborations among users and applications in the Cloud, and delivers Cloud services via sensor-rich mobile devices. BodyCloud also offers decision support services to take further actions based on the analyzed BSN data.

The BodyCloud approach is centered around four main decentralized components (or sides): Body, Cloud, Viewer, Analyst. The *Body*-side is the component, currently based on SPINE Android [23], that monitors an assisted living through wearable sensors and stores the collected data in the Cloud by means of a mobile device. The *Cloud*-side is the component, currently implemented atop Google App Engine [33], that provides fully support for specific applications through data collection, processing, analysis and visualization. The *Viewer*-side is the Web browser-enabled component able to visualize the output of data

analysis through advanced graphical reporting. The *Analyst*–side is the component that supports the development of BodyCloud applications.

The core of the BodyCloud architecture is the Cloud-side that offers high-level Web-based programming abstractions for the rapid prototyping of Cloud-assisted BSN applications: *group*, *modality*, *workflow*, and *view*. A group formalizes a specific application which manipulates a specific BSN data source. A modality formalizes a specific interaction between the Body-, Cloud- and Viewer-sides within a group. It specifies the input data, the actions to be performed on the input data, and the output data. A workflow formalizes an analysis process, producing output data from input data; it is specifically composed of one or more *nodes* (organized in an acyclic graph) representing specific algorithms. A view formalizes the graphical visualization of output data for the Viewer-side.

The proposed SaaS approach is, to the best of our knowledge, the first general-purpose software engineering approach for Cloud-assisted community BSNs. It notably allows for rapid prototyping of Cloud-assisted BSN applications, customizability of the architectural components through Web standards-based procedures, and scalability due to the employed Google App Engine PaaS infrastructure.

A case study for electrocardiography (ECG) is presented to highlight the effectiveness of BodyCloud in supporting the development of Cloud-assisted BSN applications. An application based on the *ECG as a Service* allows to collect, process, store and analyze ECG data streams generated by sensors worn by several individuals.

The rest of the paper is structured as follows. Section 2 discusses the motivations and challenges of the integration of BSNs and Cloud computing. Section 3 presents the overall BodyCloud architecture, details its main programming abstractions, and highlights the distinctive properties of the BodyCloud approach. Section 4 presents *ECG as a Service*, the case study developed atop BodyCloud. Section 5 discusses some related work. Finally, conclusive remarks, lessons learned and directions of future work are outlined.

2. BSN and Cloud Integration: Motivations and Challenges

The integration of BSNs and Cloud computing can provide evident benefits in the following four aspects.

- *Management*: Data management in BSNs deals with the challenging task of defining how BSN-originated data are efficiently collected, managed, stored and conveyed for processing. Activities related to the acquisition and management of data feeds from numerous body sensors in real-time may be

distributed in time and/or space [21]. Time distribution refers to activities taking place at different times, while being coordinated to have a coordinated effect. Space distribution means that activities may take place at different locations, while they are connected by a data network. A Cloud computing infrastructure can facilitate the management of distributed data and support advanced functionalities such as data fusion.

- *Processing*: The collected data from BSN nodes are processed into physical measurements and combined into composites, e.g. combining body temperature readings and blood pressure into a health chart for a particular patient. In the presence of numerous incoming data streams from a BSN, processing of data to make critical decisions in real-time requires fast processing that may be computing/resource intensive. Harnessing the computational resources of a Cloud infrastructure can be performed for the required provisioning of computing resources.
- *Service invocation*: Processed data from BSNs is associated with meaning, confidence and quality information. Specifically, the data is associated with information on how it was processed (derivation), for whom and why it was collected (agency), and how it may be distributed (rights). This process is performed through automatic formation of workflows and invocation of services. Such operational flow requires a platform to support automatic workflow formation and service invocation, potentially through a Cloud infrastructure.
- *Analysis*: Annotated BSN datasets are imported into analysis tools and modeling is performed for further use in various applications and decision making systems. The analysis operations along with the enrichment hierarchy depend on suitable storage and middleware technologies to perform highly swift data processing. It can be availed by exploiting the processing power of Cloud to provide quick response.

2.1.Body Sensor Networks

BSNs rely on the feasibility of wearing very small bio-sensors on the human body that are comfortable and that do not impair normal activities. The sensors worn on the human body collect various physiological changes to monitor a patient's health status, regardless of their location and activity. The information is transmitted wirelessly to an external processing unit. This device in turn instantly streams all information in real-time to doctors who are responsible to the patient. In the face of an emergency, physicians can provide instructions in the form of messages or alarms to the patient through a computer system. It can be considered a breakthrough improvement in healthcare and will be the cornerstone for telemedicine as it comes into fully comprehensive existence. Examples of the use of

BSNs include recognizing activities through arm/leg/waist-worn accelerometer/gyroscope sensors, detecting heart attack events by measuring changes in a patient's vital signs, specifically the ECG (electrocardiogram), auto-injecting insulin with an implanted pump as soon as the patient's insulin level declines, detecting emotions or human statuses (such as fear, stress, happiness) through EMG (electromyography), GSR (galvanic skin response) and ECG sensors.

Practical applications of BSNs aim at improving the quality of life by enabling continuous and real-time non-invasive medical assistance at low cost [36, 47]. Applications where BSN can be useful include early detection or prevention of diseases, e.g. heart attacks, Parkinson, diabetes, asthma; elderly assistance at home, e.g. fall detection, pills reminder, activity monitoring; e-fitness, e.g. calorie expenditure, posture/gesture correctness; rehabilitation after surgeries, e.g. knee or elbow rehabilitation; motion and gestures detection for interactive gaming; cognitive and emotional recognition for driving assistance or social interactions; and medical assistance in disaster events, e.g. terrorist attacks, earthquakes, bush fires. In the following we provide a brief description of some application areas of BSNs:

- *Health monitoring*: In a health-care scenario, BSN-based systems allow monitoring health and motion information in real-time. The collected data is transmitted wirelessly to a local or remote diagnosis and storage service for processing and display [30, 40, 45, 49]. Analysis of the sensed data can be performed in real-time or offline by doctors depending on the need [44].
- *Sports performance/fitness monitoring*: Wearable motion sensors can be used to monitor and analyze physiological data to gauge the movement of sports' players [6]. Such sensors may be worn at both hands and elbows or around the chest. After analysis of the data, features are extracted to assist users in exercise routine and achieve desired performance goals.
- *Interactive games*: Motion sensor-based games such as Nintendo Wii use body sensors to enable gamers to perform actual body movements, such as sword fighting, boxing, shooting. Gamers' movement information is feedback to the gaming console that processes it to deliver an interactive gaming and entertainment experience.
- *Information sharing*: Business applications can make use of body sensors to improve customers' shopping experience. Information collected from body sensors can be transmitted to a processing server to provide recommendation to the customers based on their profiles and preferences.

- *Secured authentication*: BSN can also be used in physiological and behavioral biometrics schemes such as facial pattern, iris recognition and finger prints. This can increase the level of provided security in commercial settings, by exploring the physical/behavioral characteristics of the human body.

In Figure 1, a BSN reference network and software architecture [12] is portrayed. In particular, the architecture is organized into multiple wearable sensor nodes and one coordinator node. The coordinator manages the on-body sensors, collects, stores and analyzes the data received from the sensor nodes, and act as a gateway to connect the BSN with a remote control center for coordination and monitoring. Sensor nodes measure biophysical parameters (e.g. heart pulses, ECG, body movements) and send raw or pre-processed data to the coordinator.

The software architecture of the system consists of two main components, implemented, respectively, on the coordinator (e.g. a notebook or a PDA/smart-phone) and on the wearable sensor nodes (e.g. inertial and bio sensors). The coordinator side includes three layers: User applications, Coordinator and Comm(unication) Adapter. The Coordinator layer allows registered User applications to be notified of the following events generated by the wearable sensors: discovery of new nodes, data coming from sensor nodes, node alarms, and system messages such as low battery warnings. Commands issued by user applications and sensor network-generated events are both coded in lower-level messages and decoded in higher-level information by the Comm Adapter layer according to a specific application-level over-the-air protocol. This component handles packet generation and retrieval and is interfaced with specific software components of the host platform to access the physical radio module to transmit/receive packets to/from the wearable sensors.

At the sensor node side, the software architecture provides abstractions of hardware resources such as sensors and the radio, a default set of ready-to-use common signal processing functions and, most importantly, a flexible and modular architecture to be customized and extended to support new physical platforms and sensors and introduce new signal processing services. In particular, the Comm(unication) Manager acts as the counterpart of the Comm Adapter. The Sensor Manager manages the sensors on the node platform, providing a standard interface to the diverse sensor drivers. It is responsible of sampling the sensors and storing the sensed data in properly defined data buffers. The Node Manager is the orchestrating component, responsible for interpreting the remote requests and dispatching them to the proper components. Finally, the Processing Manager consists of a dispatcher for the actual processing services and a standard interface for user-defined services integration.

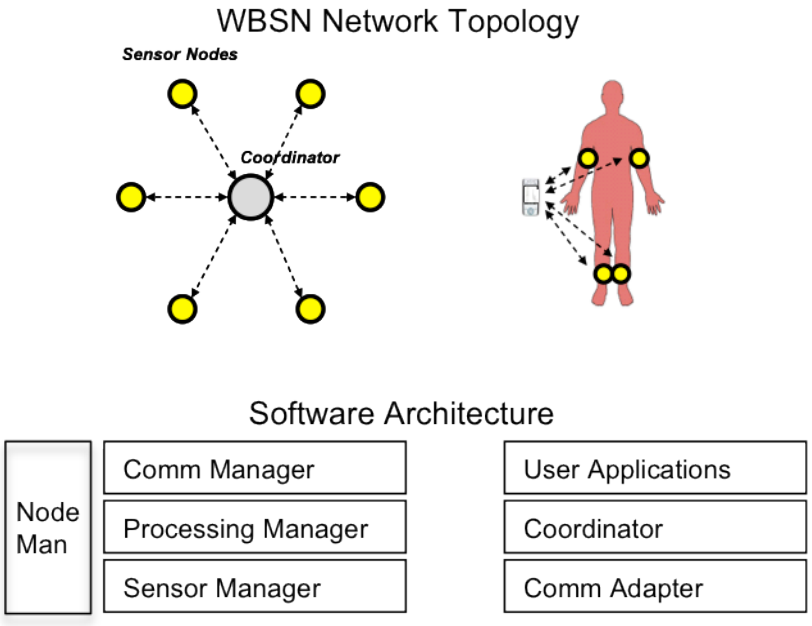


Figure 1. A BSN reference architecture.

2.2.Cloud computing ecosystem

Cloud Computing provides flexible, robust and powerful storage and computing resources, which enables dynamic data integration and fusion from multiple data sources. In addition a Cloud-based approach can offer flexibility and adaptability in the management and deployment of data analysis workflows. The dynamic deployment of software components as Cloud services eliminates the need for new client applications to be developed and deployed when the user requirements change. This also introduces an intrinsic competitive environment for the development of better services.

Cloud computing layers (IaaS, PaaS, SaaS) and software components (e.g., databases, data mining workflow tools) can be customised to support a distributed real-time system for the monitoring and analysis of BSNs data streams. Figure 2 shows the diagram of the Cloud computing ecosystem. The Cloud Computing Provider exports the Infrastructure (IaaS) integrated with a Data Mining development environment as a Platform as a Service (PaaS) to the Application Workflow Developer. The Workflow Developer deploys a particular application as Software as a Service (SaaS) to the End User (e.g. the cardiovascular biologist collecting data of many patients or the medical staff at the point-of-care). The front end of the application can be developed, for example, for a mobile device to ensure mobility and portability. The approach is based on the customisation of an open-source Cloud computing toolkits using

Cloud Computing standards [11] and integrated with data mining development tools and workflow management systems (e.g. KNIME [13], RapidMiner [42], Weka [28]).

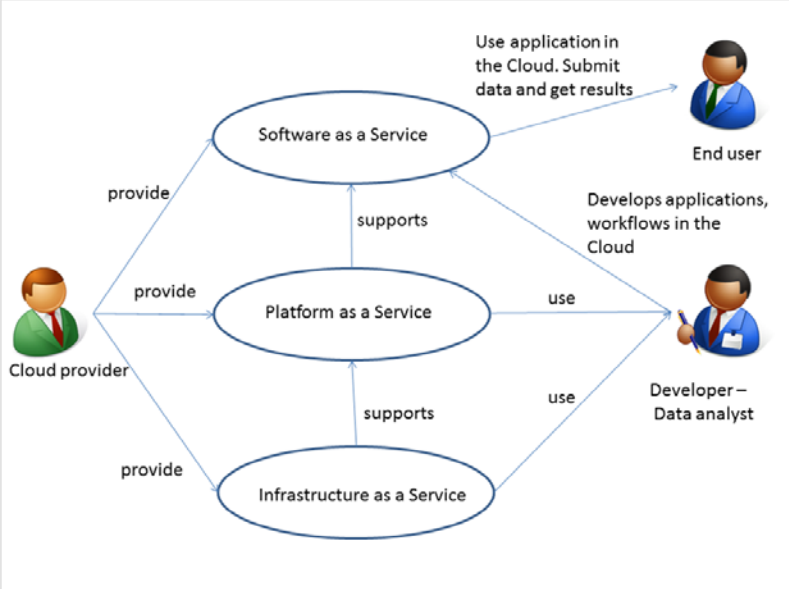


Figure 2. The Cloud computing ecosystem.

2.3.Integration challenges

While there are main advantages of BSN adoption in various applications, there are a number of associated challenges that need to be addressed [29]. Moreover, integration of BSN with a Cloud infrastructure poses additional challenges related to data management, system implementation and real-time computing. In the following we first list BSN-related challenges, followed by specific challenges regarding the BodyCloud system that integrates BSN with Cloud to perform effective data stream processing.

- *Interoperability*: A BSN system requires ensuring seamless data transfer across different standards to promote information exchange, plug-and-play device interaction and uninterrupted connectivity.
- *Heterogeneity*: A BSN system should be capable of integrating various different sensors in terms of complexity, power efficiency, storage, and ease-of-use. Moreover, it should provide a common interface between the sensors and a storage service to facilitate remote storage and viewing of sensed data as well as access to external processing and networked analysis tools.

- *Security*: Transmission of BSN data streams should be secured to prevent potential intruders. Moreover, integrity of each patient's data has to be maintained with guarantee that one patient's data is not mixed with another patient's data.
- *Protecting privacy*: One key concern of BSN users is to protect the privacy of personal data. A BSN system should ensure that patient's privacy is maintained even when data is being analyzed using an external tool. In addition, social awareness and acceptance is required for wider applications of BSN.
- *Data validation and consistency*: Data collected from multiple sensor nodes need to be collected and analyzed in a seamless fashion. BSN sensors are subject to inherent communication, hardware and network failures that may result in erroneous datasets. It is crucial that the sensed data is validated and data quality is maintained to reduce any noise in the data and identify possible weakness in the infrastructure.
- *Interference reduction*: BSN mostly uses wireless connectivity for data communication. The wireless link should be able to reduce interference and increase the co-existence of sensor nodes with other networked devices. This is to ensure that the functionalities of BSN nodes are not degraded due to the presence of other devices capable of possible interruption in seamless data transmission.

In addition to the above-mentioned BSN challenges, integration of BSN with a Cloud computing infrastructure to develop the BodyCloud system poses the following research challenges:

- *Complex event processing and management*: Real-time data streams from BSN may trigger certain events and services in the Cloud. These data streams are analyzed and results are used in applications for decision making by identifying contextual and situational information.
- *Massive scale and real time processing*: Integration of heterogeneous BSN generating vast amounts of data is a challenge, especially in the presence real time requirements. If a BSN is used to generate real-time multimedia content such as streaming video, audio and images, it poses additional challenge to accurately process and store the data in a Cloud environment.
- *Large scale computing frameworks*: The allocation of computational and storage resources as well as data migration in the Cloud is critical when multiple BSN data sources are not co-located. This is particularly challenging when the data sets and their corresponding access/search services are geographically distributed within the Cloud.
- *Harvesting collective intelligence*: While heterogeneous and real-time BSN data feeds allow improving decision making by using data and decision level fusion techniques, to maximize the intelligence that can be exploited from massively co-located information in the cloud is a challenge.

- *Data stream management*: Data management will include data format conversion into standard formats, data cleaning and aggregation to improve data quality, and data transfer to storage Clouds.
- *Interfacing the Cloud with BSN*: There should be an interface between BSN resources and the Cloud fabric. Communication interfaces are required to manage network connectivity between BSN and the Cloud. BSN nodes will be exposed as Cloud services and indexed via indexing services. There also has to be provision to manage sensing jobs and data from sensor networks. Virtualization will be a key technology. The proposed is framework provides various services for the underlying sensor resources, such as power management, security, availability, and QoS.

3. A SaaS Architecture for Ubiquitous Body Sensor Network Applications

BodyCloud is an architecture for the integration of BSNs and a Cloud PaaS infrastructure. The system architecture and its basic components and services are shown in Figure 3. The design of the general architecture follows some requirements to support sensor data management, concurrent application execution, mashup service invocation and data analysis:

- Provide functionality to receive and manage sensor data in a highly seamless manner from a BSN.
- Set up a scalable framework to support the processing of multiple data streams for concurrent application services.
- Persistent storage and exchange of sensor data and analysis results to enable further decision making.
- Reuse of a PaaS infrastructure, thus providing a SaaS level approach.

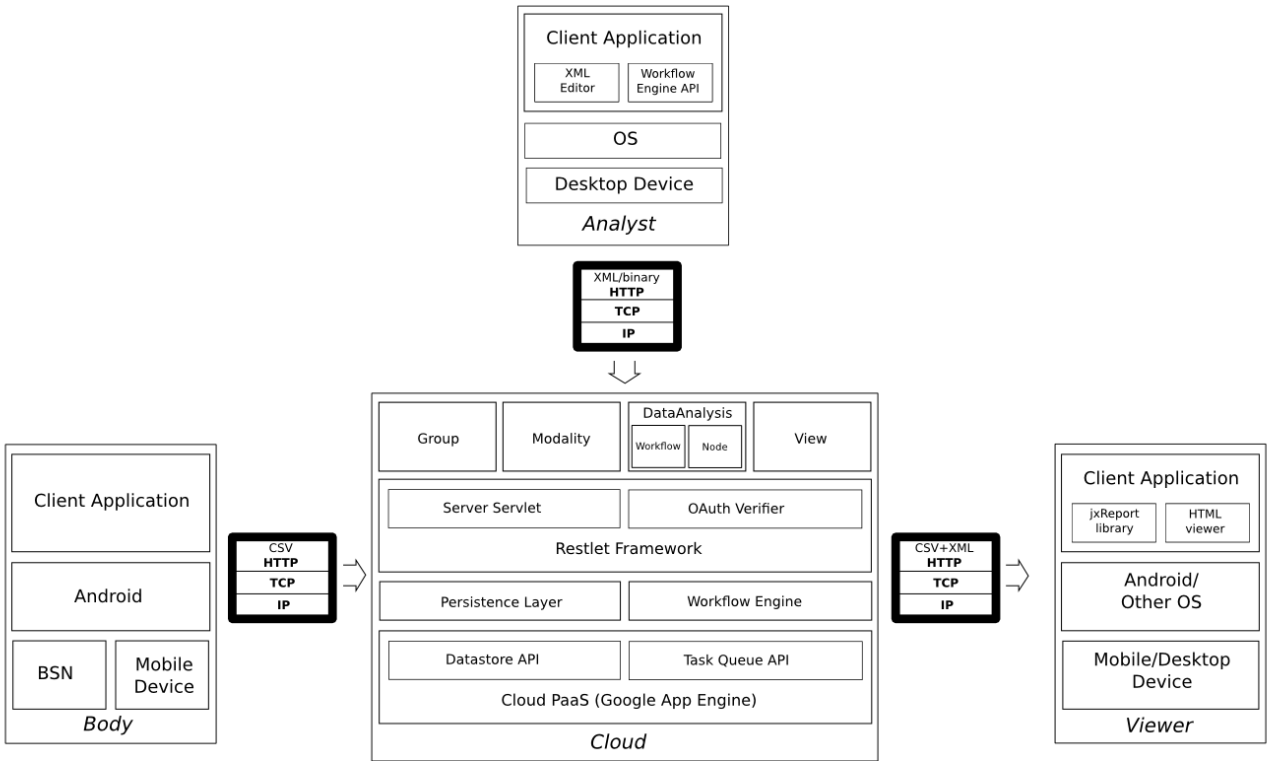


Figure 3. BodyCloud architecture.

In particular, the architecture is composed of four main subsystems (or sides):

- *Body* is the system side that monitors the assisted living by means of wearable sensors and sends the collected data to the Cloud, e.g. via a mobile device.
- *Cloud* is the system side that provides full support for specific applications through data collection, processing/analysis and visualization. Specifically, each specific application can be defined in terms of *Groups*, *Modalities*, *Workflows*, *Nodes*, *Views*.
    - *Groups:* a Group formalizes a specific application which manipulates a specific BSN data source.
    - *Modalities*: a Modality formalizes a specific interaction between Body, Cloud and Viewer, within a Group. It specifies the input data, the actions to be performed on the input data, and the output data.
    - *Workflows*/*Nodes:* a Workflow formalizes a data-flow process that analyzes input data to generate output data. A *Workflow* is composed of one or more Nodes organized in a directed acyclic graph. Nodes represent specific algorithms and links between nodes are data flows.

    - *Views:* a View formalizes the visualization of the output data for Viewers.
- *Viewer* is the system side able to visualize the output of data analysis through advanced graphical reporting.
- *Analyst* is the side of the system that supports the development of new BodyCloud applications.

The abstractions of the Cloud-side are supported by a RESTful web service, implemented using the Restlet Framework, making the interaction with the Cloud-side HTTP based. Requests are handled by a special servlet provided by the framework and dispatched according to its URI. Each request must include the HTTP authorization header. The authorization system is based on OAuth 2.0 [31] using the access token as authorization string [34], which is validated by the OAuth Verifier and used for user authentication. Requests are handled synchronously but, as analysis processes can take a long time to be computed, the web service also provides the capabilities to queue them for asynchronous execution.

The application business logic is built around an object oriented Domain Model [25] of the Cloud-side abstractions, as shown in Figure 4. The abstract class Entity identifies a concept within the abstraction. Each instance of Entity has a type, represented by its concrete class, a name that identifies the Entity among the others of the same type and an UUID (Universal Unique Identifier) that univocally identifies the object among all other entities. The underlying Persistence Layer deals with entities persistence: it is accessed by the web service through a Java interface which keeps the data source completely independent.

The Workflow Engine is the component that is delegated of workflows execution. Similarly to the persistence layer, it is an independent component, accessible through interfaces. The design of the Workflow Engine interfaces facilitates either the implementation of ad-hoc data analysis applications or the adoption and integration of third party data mining workflow software (e.g., e.g. KNIME [13], RapidMiner [42], Weka [28]). In the case study presented in this work, an ad-hoc implementation of a simple Workflow Engine has been used. This implementation supports processing tasks pipelines and workflow definition is based on XML.

BSN data (gathered through the web service) and analyzed data (output of the workflow engine) are both encoded as a table (data), each column representing a particular variable.

The SaaS layer relies on the Google App Engine (GAE) PaaS, which provides a Servlet Container and Cloud Computing services. The Datastore API allows the application to access GAE scalable datastore

and is used to implement the persistence layer, while the Task Queue API is used by the Web service (through a Java interface) to queue requests.

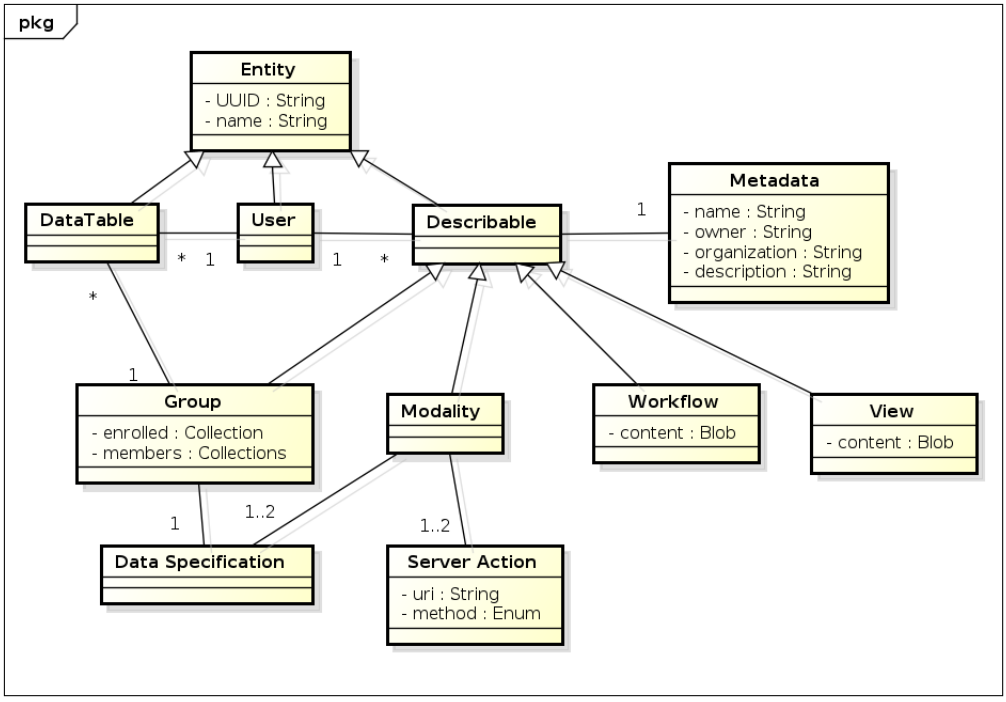


Figure 4. Domain Model

The interactions among the BodyCloud sides can be of the following types.

- The interaction between Body and Cloud (see Figure 5), is formalized in three steps:
    1. Retrieve the list of all available modalities, making an HTTP GET request a standard URL. Cloud will reply with an XML document containing a list of available modalities, their descriptions and URLs;
    2. Select desired modalities and retrieve their specifications via the associated URLs;
    3. As new data are generated by Body, they are sent to Cloud by means of an HTTP PUT request to a specific server resource that refers to a specific group, as specified in the selected modality.
- The interaction between Viewer and Cloud (see Figure 6), is formalized in four steps:

1. Retrieve the list of all available modalities, making an HTTP GET request to a known URL. The server will reply with an XML document containing a list of available modalities, their descriptions and URLs;
2. Select desired modalities and retrieve their specifications via the associated URLs;
3. Make an HTTP POST request to a specific Cloud resource that refers to the specific analysis process (workflow), as specified in the modality. The request must contain all the required parameters such as target user and group, encoded as post form. As soon as Cloud performs the task, the viewer receives the output data, encoded as CSV;
4. Download a view, making an HTTP GET request to the view URL specified in the modality.

- The interaction between Analyst and Cloud (see Figure 7) involves:
  1. Defining a new *workflow*;
  2. Defining a new *group*;
  3. Defining a new *view*;
  4. Defining a new *modality*.

  Each definition consists of an XML document describing a new resource, which is uploaded by means of an HTTP PUT request, using the resource type and name as endpoint.

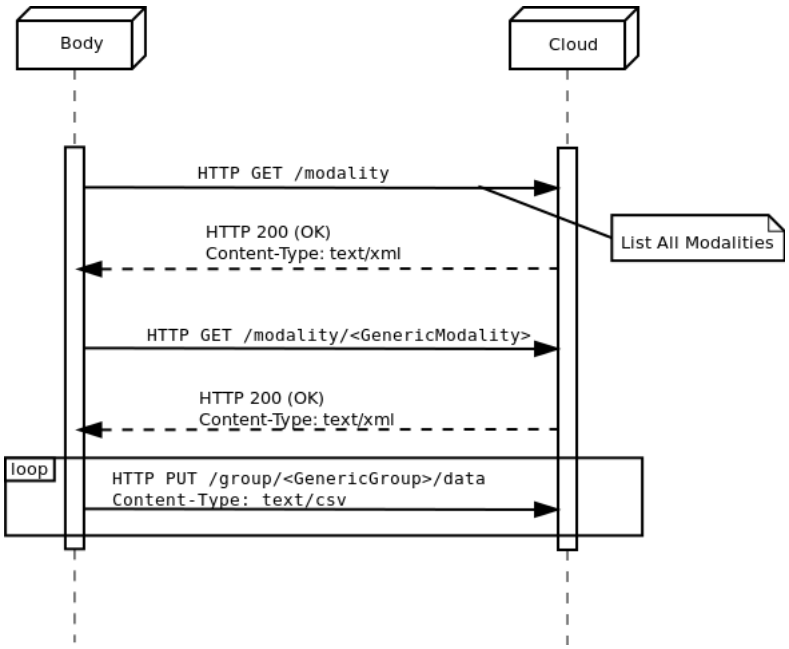


Figure 5. Basic interaction beween Body and Cloud

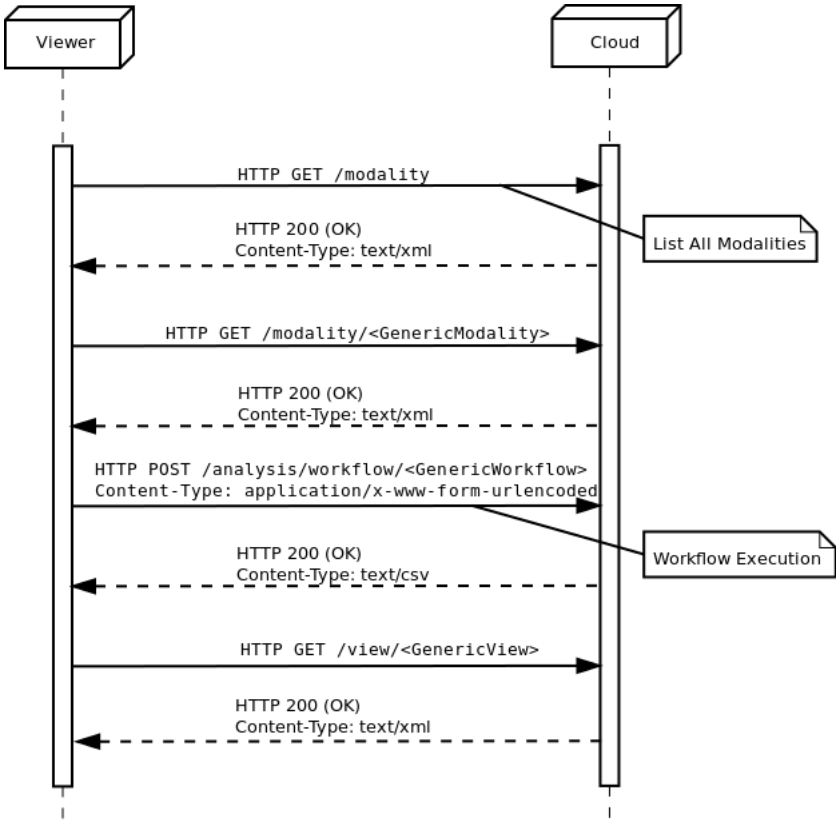


Figure 6. Basic interaction beween Viewer and Cloud

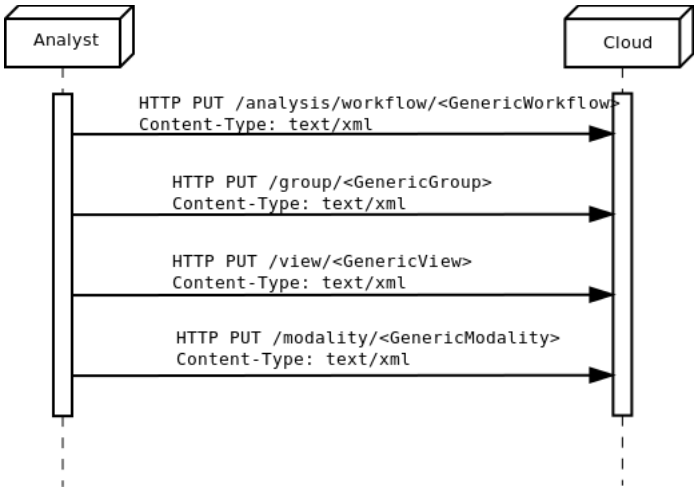


Figure 7. Basic interaction beween Analyst and Cloud

In the following subsections the architectural components of BodyCloud are described in detail.

3.1.Body-side: data acquisition

Data acquisition is currently based on Android-SPINE, the Android version of the SPINE middleware [23]. It allows Android-enabled smartphones and tablets to be used as coordinator of the BSN and, at the same time, as Body-side of the BodyCloud platform. In particular, data collected through Android-SPINE are then streamed up to the Cloud-side, using the real-time data feed modality (see next subsection). In Android-SPINE wearable sensors communication is based on Bluetooth. The prototype implementation fully supports Shimmer nodes. The following functionalities are provided by the application-level SPINE protocol [12].

- *Sensor Discovery* allows for the discovery of wearable sensor nodes and their supported sensing and processing capabilities.
- *Sensor Configuration* allows the selection of specific sensors in a sensor node and the specification of their parameters (e.g., the sampling rate).
- *In-node Processing* allows to compute one or more functions on the collected sensor data and to send the merged data streams (if more than one) to the base station. In particular, each function is computed on a data window according to a given shift, defined as percentage of the data window. For each node, it is possible to enable one or more functions independently. The processing functions may be activated over-the-air via additional specific parameters.

- *BSN activation(deactivation)*, which allows to control the synchronized start-up (or stop) of the activity of the BSN sensor nodes.
- *Data collection* allows to gather raw or preprocessed data coming from the sensors, at the coordinator.
- *Logging* allows to archive and visualize log messages produced by the sensor node for debugging purposes, at the coordinator.

It is worth noting that the Body-side could use different BSN middlewares/frameworks [12, 23].

3.2.Cloud-side: programming abstractions

The Cloud-side is based on the programming abstractions *group*, *modality*, *workflow* and *view*, as shown in the domain model of Figure 4. Such programming abstractions are implemented as HTTP resources and their representations (encoding of the current or intended state of the resource), thus exposing a Web API.

On each resource one or more HTTP methods can be executed, their meaning is analogue to the HTTP/1.1 standard [22], as shown in Table 1.

Table 1: HTTP Methods

GET	Retrieves the current representation of the resource.
PUT	Uploads the intended representation of the resource. If the resource doesn't exist, it will be created. The user who created a resource is its owner, the only one who can edit or delete it.
POST	Requests that the server accepts the entity enclosed in the request as a new subordinate of the resource. The enclosed entity is encoded as post form.
DELETE	Deletes the resource. A resource can be deleted only by its owner.

In the following subsections each programming abstraction is described in detail.

3.2.1 Group

A group is defined by three correlated resources (collector, data, contributor) described respectively in Tables 2-4.

Table 2: Group Collector

DESCRIPTION	A data collector is intended to gather Group Data (e.g., data tables) which comply to the same specification.
URL STRUCTURE	`https://bodycloud.appspot.com/group/<name>/collector` *name* is the unique name of the group
SUPPORTED METHODS	GET, PUT, DELETE

REPRESENTATION	XML
	Specifies the data specification to be accepted (see Figure 9)
	An example:
	`<dataSpecification>`
	`<data>`
	`<name>ECGShimmerSample </name>`
	`<type>INTEGER</type>`
	`<source>ECGShimmerSensor</source>`
	`</data>`
	`<dataSpecification>`

Table 3: Group Data

DESCRIPTION	Data is a group subresource that represents the actual data collected by the group. Data is then grouped on a per user basis.
URL STRUCTURE	`https://bodycloud.appspot.com/group/<name>/data` *name* is the unique name of the group
SUPPORTED METHODS	GET, PUT, DELETE
REPRESENTATION	A table of data encoded according to the HTTP Content-Type - CSV (with header record) for `text/csv` - Arff for `text/plain` - Json for `application/json`

Table 4: Group Contributor

DESCRIPTION	The contributor is a group subresource that contains the users who uploaded data to the specified group.
URL STRUCTURE	`https://bodycloud.appspot.com/group/<name>/contributor`

	name is the unique name of the group
SUPPORTED METHODS	GET

REPRESENTATION	An XML document with the identity of the users
	An example:
	`<users>`
	`<user>user1@gmail.com</user>`
	`<user>user2@gmail.com</user>`
	`</users>`

3.2.2 Modality

A modality encodes a body-cloud or a viewer-cloud interaction and is intended to be interpreted and executed by a client application (see Table 5).

Table 5: Modality

DESCRIPTION	A modality formalizes a specific interaction between Body, Cloud and Viewer
URL STRUCTURE	`https://bodycloud.appspot.com/modality/<name>` *name* is the unique name of the modality
SUPPORTED METHODS	GET, PUT, DELETE

REPRESENTATION	XML
	`<modality>`
	`<inputSpecification/>`
	`<init-action/>`
	`<action/>`
	`<outputSpecification/>`
	`</modality>`

Each modality defines a specific service, such as data feeds, data analysis tasks, single-user or community applications, etc. A modality defines the specifications of input and output data format, protocols for data transfer, the flow of processing tasks to transform input data into output data and the specifications of output data visualization. Modalities can be activated individually and in groups to provide a service to the user.

A modality is defined according to the XML schema portrayed in Table 5. The data input and output specifications, `inputSpecification` and `outputSpecification`, are needed if the

modality can, respectively, receive and send data. The tags `init-action` and `action` are used to specify Web API calls. In particular, the tag `action` is used to specify an operation that sends input data and/or retrieves output data to/from a Cloud-based workflow, according to `inputSpecification` and `outputSpecification`. The tag `init-action` is optional and can be used to initialize a resource or to retrieve data needed for the execution of an `action`.

The tags `inputSpecification` and `outputSpecification` are data specifications defined according to the XML schema `dataSpecification` reported in Figure 8. The tag `dataSpecification` encodes the header of a dataset, includes one or more `data` sections and an optional `view`. The tag `data` represents a single dataset column and is specified by a `name`, which a `type` (e.g. `DOUBLE`, `INTEGER`, `TIMESTAMP`) and (optionally) a `source`, which specifies the source sensor to be used to gather data (e.g. `HEARTBEATER`, `CLOCK`, `TERMOMETER`, `ECGSENSOR`). The tag `view` is for visualization purposes and can be used to specify the URI of the view specification.

```
<dataSpecification>
 <data>
  <name/>
  <type/>
  <source/>
 </data>
 <view/>
</dataSpecification>
```

Figure 8. XML schema of dataSpecification

The tags `init-action` and `action` are defined according to the schema reported in Figure 9. The tag `uri` is the relative URI to be used for the action request. The tag `method` is the HTTP method to be used for the request. The tag `repeat` (optional) can be either true or false (default) and specifies whether or not the action must be repeated. The tag `trigger` (optional) has some XML attributes, which define the trigger that activates the action. Currently the only trigger supported is `after`, which activates the action after a specified time (in seconds). The tag `parameter` is only valid for POST requests and specifies a parameter <`name`, `value`> for POST forms. The value of the parameter can also be taken from an external XML code using an XPath expression; in this case, the tag `reference` is used instead. The tag `reference` has two attributes: `xpath` specifies the expression and `type` defines how the expression output must be used.

```
<action>
 <uri/>
 <method/>
 <parameter>
   <name/>
   <value/>
   <reference xpath=”” type=””/>
 <parameter/>
 <repeat/>
 <trigger/>
</action>
```

Figure 9. XML schema of action

The execution of a modality is carried out as follows:

- The client (Body-side or Viewer-side) starts collecting data according to `inputSpecification`, whether it is specified. This is a typical Body-side scenario.
- The `init-action` is intended either to prepare the server for the proper `action` or to retrieve run-time information. The optional XML output of such action must be stored. The `init-action` can be executed at the beginning or when triggered if a trigger is specified. Any parameter of the `init-action` cannot include the `reference` tag. The `action` must be executed after the `init-action` (if any) and when triggered (if a trigger is specified). The parameter reference should be resolved as follows:
    - use the `init-action` XML output to evaluate the XPath expression;
    - if the reference is of type `CHOICE` (default), then let the user choose between the expression results;
    - If the reference is of type `MAP` a different request should be made using each of the expression results.

  The action output must match the `outputSpecification` (if it is present, otherwise must be empty). The client must also fetch the view specification according to the URI enclosed in the tag `view` and generate the report (this is a typical Viewer-side scenario).

3.2.3 Workflow

A workflow describes a set of operations the server engine should perform (see Table 6).

Table 6: Workflow

DESCRIPTION	A workflow is pipeline of operations to perform on data.
URL STRUCTURE	`https://bodycloud.appspot.com/engine/workflow/<name>` *name* is the unique name of the workflow
SUPPORTED METHODS	GET, PUT, POST, DELETE

REPRESENTATION	Workflow XML schema
	`<workflow>`
	`  <node>`
	`  <type/>`
	`  <parameter>`
	`  <name/>`
	`  <value/>`
	`  </parameter>`
	`  </node>`
	`</workflow>`

Worlflows in BodyCloud are direct acyclic graphs (DAGs). A node (see Table 7) may have static and dynamic parameters. Static parameters are defined directly in the workflow XML schema, whereas dynamic parameters must be supplied at runtime through the web service. Nodes can be developed as Java code from the workflow engine library (see Figure 3). The library contains the `Node` interface, which every node must implement, utility classes, and a simple testing environment. Once implemented, the node can be packed within a jar and uploaded to the Cloud-side, where it can be used in workflows. A node should only perform operations on data, but BodyCloud also provide three built-in nodes for accessing the datastore:

- `UserDataReader` reads data given a user and a group. The dynamic parameters are `sourceGroup` and `sourceUser.`
- `GroupDataReader` reads all data associated to a given group. It takes `sourceGroup` as dynamic parameter.
- `UserDataWriter` writes data to a group by username. The dynamic parameters are `destinationGroup` and `destinationUser.`

Table 7: Node

DESCRIPTION	Node of the back-end engine.
URL STRUCTURE	`https://bodycloud.appspot.com/engine/node/<name>` *name* is the unique name of the node
SUPPORTED METHODS	PUT, DELETE
REPRESENTATION	A jar file to be sent as binary data. The Content-Type should be `application/octet-stream`. The jar must contain a Java class with the same name specified in the URL implementing the `Node` interface.

A workflow is defined according to the XML schema portrayed in Table 6. `node` is an execution unit; additional nodes can be uploaded as plug-ins. `type` is the name of a Java class implementing the `Node` interface. `parameter` is a static parameter of a node; specifically, multiple workflows could use the same node with different parameters. `name` is the name of the parameter whereas `value` is the value of the parameter. The POST method is used for workflow executions and its usage is shown in Table 8.

Table 8: Workflow Execution

DESCRIPTION	Server request that executes a workflow
URL STRUCTURE	`https://bodycloud.appspot.com/engine/workflow/<name>` *name* is the unique name of the workflow
SUPPORTED METHODS	POST
REPRESENTATION	`The Internet media type (application/x-www-form-urlencoded )` data including the dynamic parameters required by the nodes of the specific workflows; CSV data encoding the workflow output (if any)

3.2.4 View

The resource *view* specifies the visualization layout of output data and is shown in Table 9.

Table 9: View

DESCRIPTION	A view is a specification describing a graphical representation of output data.
URL STRUCTURE	`https://bodycloud.appspot.com/view/<name>` **name** is the unique name of the view
SUPPORTED METHODS	GET, PUT, DELETE
REPRESENTATION	Main part of the XML schema: <xs:element name="report" type="reportType"/> <xs:complexType name="reportType"> <xs:choice minOccurs="1" maxOccurs="unbounded"> <xs:element name="htmlTitle" type="xs:string" maxOccurs="1" minOccurs="0" /> <xs:element name="bodyCss" type="cssType" maxOccurs="1" minOccurs="0" /> <xs:element name="cssScript" type="xs:string" maxOccurs="1" minOccurs="0" /> <xs:element name="lineChart" type="lineChartType"/> <xs:element name="barChart" type="barChartType"/> <xs:element name="pieChart" type="pieChartType"/> <xs:element name="textContent" type="textContentType"/> <xs:element name="table" type="tableType"/> </xs:choice> </xs:complexType> The complete XML schema can be found here: http://code.google.com/p/jxreport/source/browse/trunk/src/schema.xsd

3.2.5 Common operations

Instances of *node*, *workflow*, *view* and *modality* can be listed as shown in Table 10. Furthermore, each of them is associated to a metadata resource, which is described in Table 11.

Table 10: Index

DESCRIPTION	This endpoint represents an index of resources
URL STRUCTURE	`https://bodycloud.appspot.com/<type>` where *type* can be group, modality or view or

	`https://bodycloud.appspot.com/engine/<type>` where *type* can be workflow or node
SUPPORTED METHODS	GET

REPRESENTATION	XML
	An index resource URLs, with another optional link to their metadata
	An example:
	`<index>`
	`<item>`
	`<reference href=”https://bodycloud.appspot.com/modality/sample”/>`
	`<metadata href=”https://bodycloud.appspot.com/metadata/sample”/>`
	`</item>`
	`</index>`

Table 11: Metadata

DESCRIPTION	A set of metadata that describes another resource
URL STRUCTURE	`https://bodycloud.appspot.com/metadata/<uuid>` *uuid* is an alphanumeric string that identifies univocally a resource
SUPPORTED METHODS	GET, PUT

REPRESENTATION	An XML with four optional text fields:
	`<metadata>`
	`<name/>`
	`<owner/>`
	`<organization/>`
	`<description/>`
	`</metadata>`

3.3.Analyst-side: data analysis

At the Analyst-side, users can create new BodyCloud applications (or services) by defining *workflows* , *groups*, *modalities*, and *views*. Each entity can be created with an HTTP PUT request to the corresponding Cloud-side resource, thus requiring only a simple HTTP client as Analyst-side supporting application. As the workflow requires new *nodes* to be developed, the Analyst-side also requires an appropriate development environment. Once developed, new nodes are also uploaded with an HTTP PUT request to the corresponding Cloud-side resource. A predefined set of nodes is typically available,

depending on the adopted implementation of the Workflow Engine. A complete service is centered around the analysis process to be performed. The service definition can be summarized as follows (see Figure 10):

1. *Algorithms*: Implementation and uploading of all required algorithms as nodes. All uploaded nodes are stored and can be used by every other user.
2. *Data Source*: Definition of a group containing the specification of data that can be gathered by the BSN and processed by the algorithms.
3. *Workflow*: Definition of the data analysis process, through the combination of existing nodes and their static parameters. The first node should read input data from the Data Source, this can be done using the built-in nodes `UserDataReader` or `GroupDataReader`.
4. *View*: Definition of one or more graphical formats of the workflow output.
5. *Modalities*: At least a Body-side specific modality and a Viewer-side specific one must be defined. The Body-side modality should have an input specification similar to the Data Source, an action that will upload the data to the group defined at point 2 and no output specification. The Viewer-side modality should perform the workflow execution as action, the parameters of which must be defined accordingly to the node documentation. Its output specification must match with the workflow output and contains the relative reference to the view.

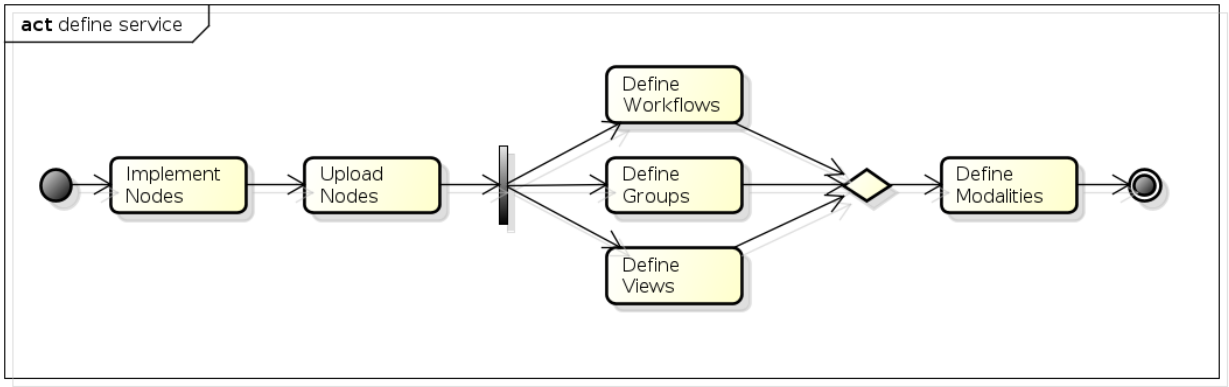


Figure 10. Service definition activity diagram

3.4.Viewer-side: data visualization

On the Viewer-side when the user requests the execution of an analysis process (workflow), the generated output is a data object (model) that complies to the `outputSpecification`. Output data are visualized (view) in the client application in a particular layout. The formatting of the output data

into a view follows the XML schema of the resource view (see Table 9). The client application applies the view specification to the model to generate the view.

As part of the current prototype, a Java library, *jxReport*[1], has been developed and integrated to the client application: jxReport provides the functionalities to generate HTML reports from an XML schema and a model. The root node of the XML specification is the report element, which contains a set of graphical elements to be drawn in the report and populated by the output data. The approach based on jxReport provides the desirable separation between the model and the view.

The model is a set of data, presented in tabular form, the view is a jxReport compliant XML document, which includes, through special tags, references to the model columns. During report generation jxReport reads then the model, from a CSV file for example, and draws the graphic elements specified in the XML document based on the model data.

The jxReport library can be used in any Java based environment (e.g. mobile or desktop). The use of HTML as formatting language has been chosen because of its portability: almost every device is able to render an HTML page by using a Web browser or an embedded HTML viewer. The prototype implementation of the client application based on jxReport includes an HTML viewer. The report can contain text, charts and tables. Every chart is customizable and the tables are sortable and filterable by using the relative text field. Table data can also be paginated. A view contains a sequence of objects, each one has a type and a set of attributes. Objects currently supported by jxReport are tables, text containers, pie charts, bar charts and line charts. Each object has some specific attributes and all objects share a set of generic attributes. The common attributes are:

- *Id*, which univocally identifies the object within the report;
- *Class*, which is used to group a family of objects with the same properties;
- *Title*, which is an optional string which will be rendered on top of the object;
- *CSS rules*, which is a list of CSS rules to apply directly to the HTML output.

Every object can be customized by using its own properties and CSS rules. The user can customize the report: for example, table column can be rearranged, text size, font and color changed, padding and margins fixed. It's also possible to add containers and other HTML objects. Additional tags can be used to format the page more easily than using CSS rules, for example <*left*> and <*right*> tags can be used to create a two columns layout without write all the CSS rules.

[1] http://code.google.com/p/jxreport/

The library *jxReport* is used to generates a single HTML document that represents the whole report. Two external JQuery plug-ins for plotting [39] and controls [54] are embedded in the generated HTML document. An example of how a report can be represented by XML code following the jxReport specifications is show in Figure 11.

```
<report>
	<barChart>
	. . .
	</barChart>
	<table>
	. . .
	</table>
	<textContent>
	. . .
	</textContent>
	<lineChart>
	. . .
	</lineChart>
	<pieChart>
	. . .
	</pieChart>
</report>
```

Figure 11. XML formatted report

3.5. Features

BodyCloud provides the following features:

- *Rapid Prototyping*. The software engineering approach of BodyCloud supports rapid prototyping of community BSN applications. Specifically, by using the provided basic programming abstractions of *group*, *modality*, *workflow*, and *view*, new applications can be quickly defined and timely deployed by means of the simple web-based interface. The flexibility of the Android-SPINE framework, enables the adoption of new sensors at the Body-side, which can be easily integrated in in a BodyCloud application by defining new XML schemas for the data streams produced by the new sensors.
- *Extensibility and Customizability*. A large volume of streaming data generated by numerous sensors can swamp even the most robust servers designed for online transaction processing applications. The BodyCloud system endeavors to be extensible and re-targetable in such a scenario. When intermediate software components or operating systems are to be changed, the SaaS system takes care of the changes in a seamless fashion with minimal disruption to the services provided to end-users already using the system.

- *Scalability*. The scalability of BodyCloud relies on its PaaS level. The current BodyCloud prototype is based on Google App Engine (GAE) [52], which provides automatic resource scaling for Web applications.

4. A CASE STUDY: ECG AS A SERVICE

The electrocardiogram (ECG) is the standard method for measuring the electrical and functional activity of the heart. The *ECG as a Service* (ECGaaS) has been developed through the BodyCloud approach and allows to monitor (collect, process, store, analyze and visualize) the ECG data coming from individuals or group of people (e.g. assisted livings, athletes, emergency teams). Figure 12 shows the architectural schema of the overall ECGaaS system.

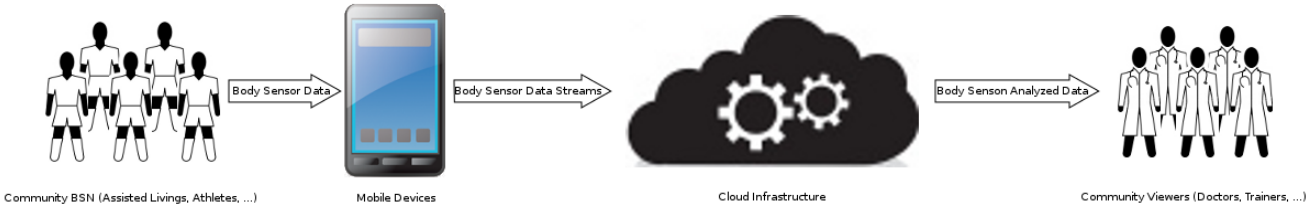


Figure 12. The ECGaaS distributed system.

A typical ECG tracing of the cardiac cycle is shown in Figure 15 and consists of a P wave, a QRS complex and a T wave. The QRS complex corresponds to the time occurrence of the heartbeat. The time interval between two consecutive R waves is called R-R interval. Traditionally, the ECG is used to diagnose cardiovascular diseases and cardiac abnormalities [56]. Recently, the ECG has also been used in the field of emotion recognition and stress detection [5]. The ECG signal is in fact a reactive signal to physiological responses due to emotion and other external factors. The advantage of using the ECG signal for detecting basic emotions is that a person can be monitored using non-invasive wearable cardiac sensors. In particular, in the proposed case study, the ECG signal is captured by the Body-side and sent to the Cloud-side in which the R-R intervals and heart rate (HR) are extracted. Two QRS detector algorithms [18] have been developed and deployed in the BodyCloud system. The first algorithm uses a fixed threshold to extract the QRS complex (heartbeat); while the second one uses an adaptive mechanism which automatically estimates the optimal threshold to extract the QRS complex from the ECG signal.

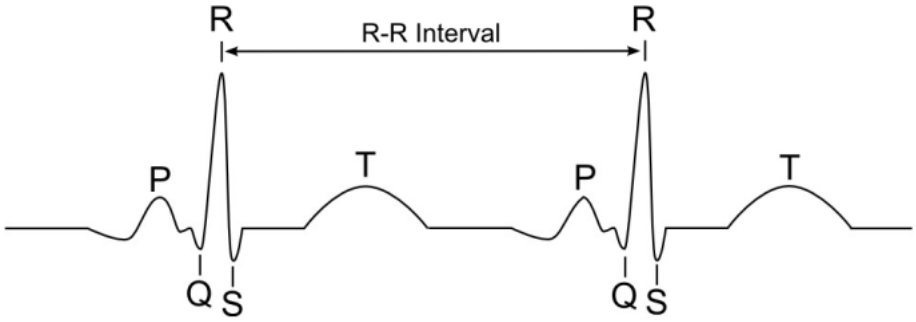


Figure 13. ECG Signal

4.1. Service programming

The specific components for the ECGaaS service are shown in Figure 14: Group, Modality, DataAnalysis (workflow/nodes) and View.

- *ECGMonitoring* represents the group of monitored users;
- *DataFeed*, *SingleAnalysis*, and *GroupAnalysis*: the former allows to transmit ECG data onto the Cloud, whereas the second and third respectively perform single and group analysis of the ECG data, specifically the extraction of the R-R signals;
- *EcgToRR* represents a workflow able to extract the R-R signal from the ECG data;
- *Tachogram* is the graphical format through which the R-R signal will be rendered at the Viewer-side.

We detail each of such components below.

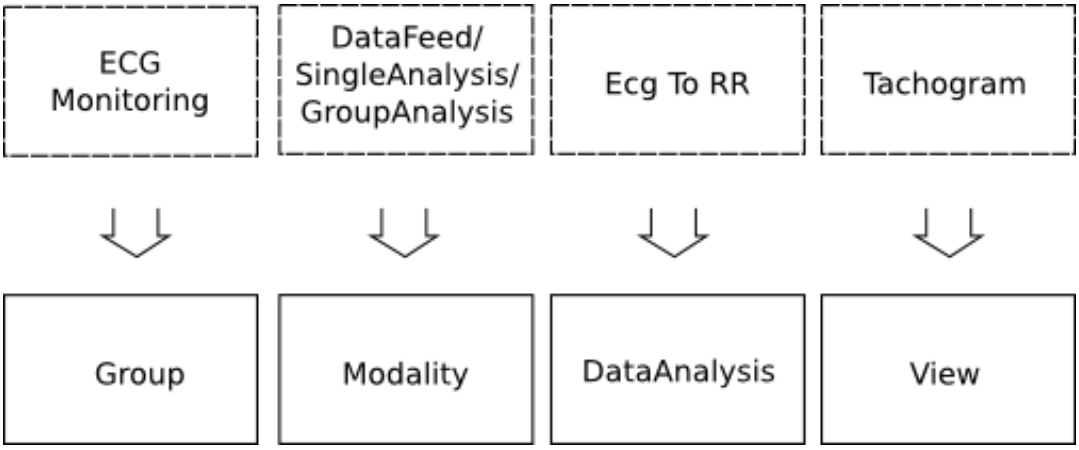


Figure 14. Body-side implemented components for the ECGaaS.

The Workflow (see Figure 15) reads ECG data associated to a specific user (through the `UserDataReaderNode` node) and calculates the R-R signal from the ECG signal by means of the `RR` node (see Figure 16).

```
<workflow>
  <node>
    <type>UserDataReader</type>
  </node>
  <node>
    <type>RR</type>
  </node>
</workflow>
```

Figure 15. Workflow

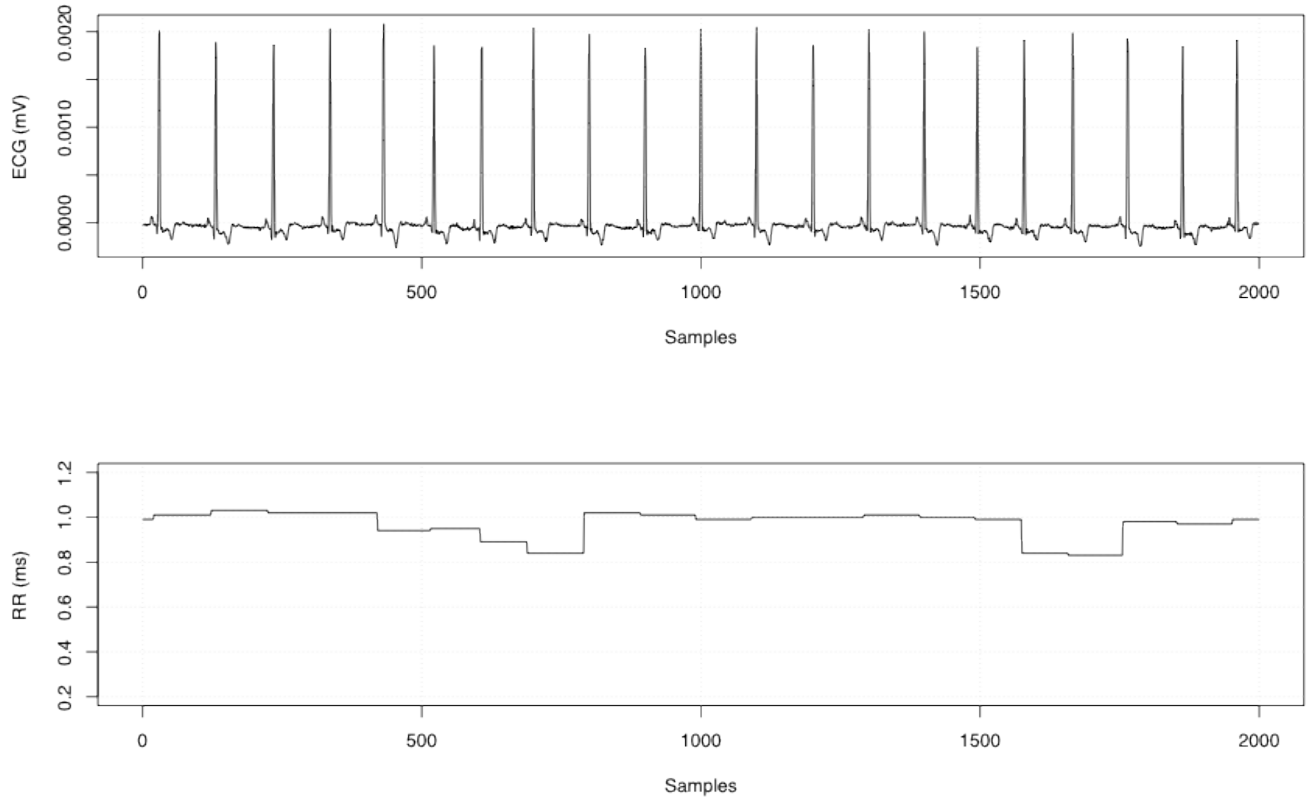


Figure 16. ECG signal and the corresponding extracted R-R signal.

The defined DataFeed modality (Figure 17) is used by the Body-side to periodically transmit the collected ECG data to the Cloud-side. In particular, the data `inputSpecification` defines the data format of the ECG sample that is gathered from the ECG Shimmer sensor board. The `init-action`

deletes any data stored previously, as the user is recording a new ECG signal. The `action` specifies that ECG data needs to be send every 60 seconds.

```
<modality>
  <inputSpecification>
    <data>
      <name>ECGShimmerSample </name>
      <type>INTEGER</type>
      <source>ECGShimmerSensor</source>
    </data>
  </inputSpecification>
  <init-action>
    <uri>/group/ecg-monitoring/data</uri>
      <method>DELETE</method>
  </init-action>
  <action>
    <uri>/group/ecg-monitoring/data</uri>
      <method>PUT</method>
      <repeat>true</repeat>
      <trigger after="60"/>
  </action>
</modality>
```

Figure 17. The DataFeed modality

```
<modality>
  <init-action>
    <uri>/group/ecg-monitoring/contributors</uri>
    <method>GET</method>
  </init-action>
  <action>
    <uri>/engine/workflow/ecg</uri>
    <method>POST</method>
    <parameter>
      <name>sourceUser</name>
      <reference xpath="//users/user"/>
    </parameter>
    <parameter>
      <name>sourceGroup</name>
      <value>ecg-monitoring</value>
    </parameter>
    <repeat>false</repeat>
  </action>
  <outputSpecification>
    <data>
      <name>rr</name>
      <type>DOUBLE</type>
    </data>
    <view>/view/tachogram.xml</view>
  </outputSpecification>
</modality>
```

Figure 18. The SingleAnalysis modality

The SingleAnalysis modality (Figure 18) calculates the R-R signal of a given target user (e.g., a patient). SingleAnalysis can be executed on the Viewer-side either by a monitoring user (e.g., a doctor or a therapist), or by the target user (at the Body-side) to monitor their own ECG. In particular, the SingleAnalysis client queries the server for selected target user, by means of a reference tag and an XPath expression. The analysis of target's data is then executed at the server and the resulting report is rendered at the client according to the tachogram specification.

The GroupAnalysis modality (Figure 19) is analogue to the SingleAnalysis, but it is designed to monitor all users within a group. The difference between GroupAnalysis and SingleAnalysis is the specification of the type parameter of the reference XPath as MAP which instructs the client to execute a different action for each value of the XPath expression (each user in this case).

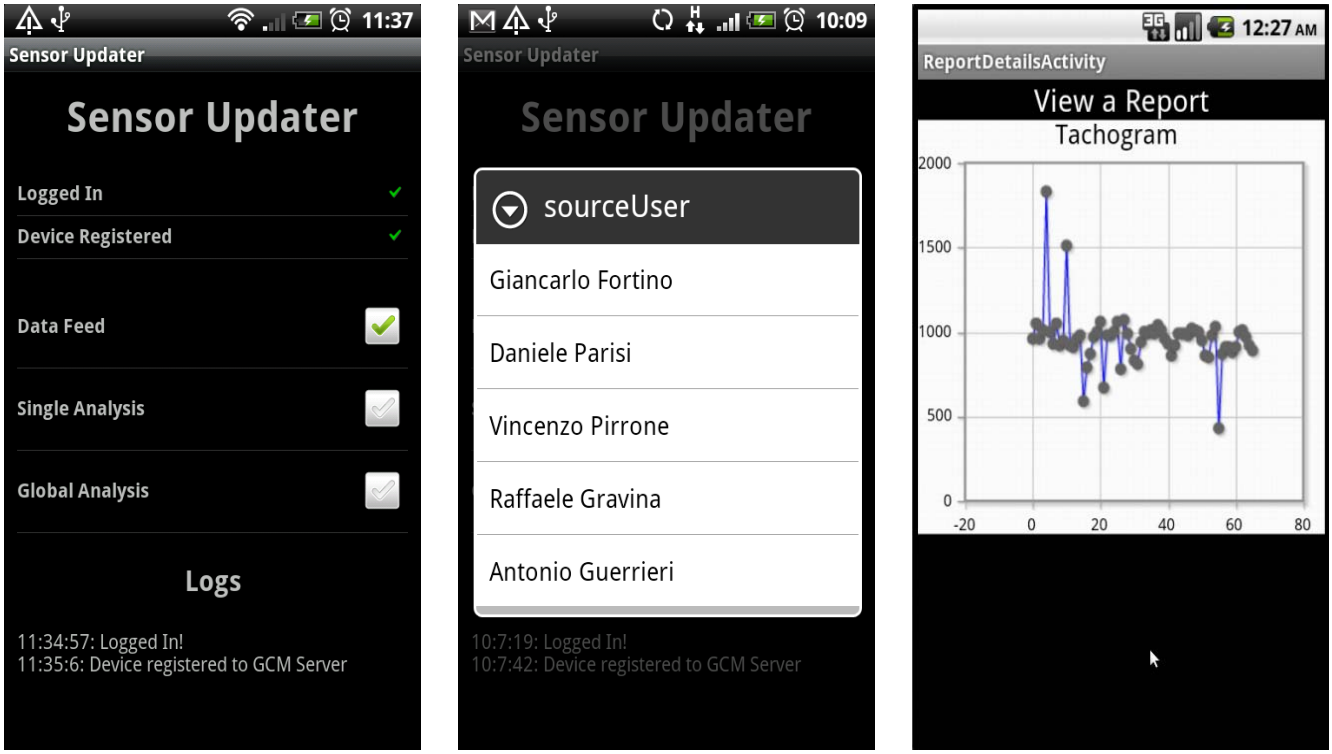
```
<modality>
  <init-action>
    <uri>/group/ecg-monitoring/contributors</uri>
    <method>GET</method>
  </init-action>
  <action>
    <uri>/engine/workflow/ecg</uri>
    <method>POST</method>
    <parameter>
      <name>sourceUser</name>
      <reference xpath="//users/user"/ type=”MAP”>
    </parameter>
    <parameter>
      <name>sourceGroup</name>
      <value>ecg-monitoring</value>
    </parameter>
    <repeat>false</repeat>
  </action>
  <outputSpecification>
    <data>
      <name>rr</name>
      <type>DOUBLE</type>
    </data>
    <view>/view/tachogram.xml</view>
  </outputSpecification>
</modality>
```

Figure 19. The GroupAnalysis modality

The view in Figure 20 encodes the tachogram, which is defined as a line chart with the R-R signal in the y-axis and a counter in the x-axis.

```
<report>
  <lineChart>
    <title>Tachogram</title>
    <showLegend>false</showLegend>
    <showMarker>false</showMarker>
    <line>
      <lineLabel>First Line</lineLabel>
      <lineColor>blue</lineColor>
      <lineWidth>1</lineWidth>
      <showLine>true</showLine>
      <points>
        <xMarker>
          <counter/>
        </xMarker>
        <yMarker>
          <index>0</index>
        </yMarker>
      </points>
    </line>
  </lineChart>
</report>
```

Figure 20. The Tachogram View



(a) (b) (c)

Figure 21. Android-based Body-side GUI: (a) source user selection list; (b) modality choice; (c) tachogram view.

4.2. Graphical User Interface

The ECGaaS client is composed of an Android-based Body-side GUI and a Viewer-side GUI which is platform-independent as it is based on HTML (see Section 3.4). Figure 21(a) shows a screenshot related to the activation of the modalities: the DataFeed modality is currently being executed. Figure 21(b) shows the screenshot for the target selection among the user of the group: the ECG of the selected target user is going to be analyzed. Figure 21(c) shows the tachogram of a user.

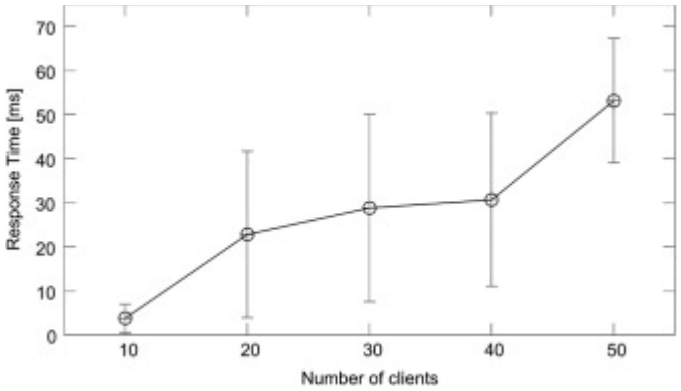


Figure 22. Scalability evaluation: DataFeed service time vs. number of clients.

4.3. Performance evaluation

A performance evaluation has been carried out by simulating a set of clients that send sensor data streams simultaneously. The ECGShimmerSensor generates approximately 6000 ECG data values a minute, so every simulated client performs 10 requests, with 60 seconds interval between each other, each one sending a dataset with 6000 samples. In order to simulate a relatively high number of clients using a reduced set of physical machines, a python multithreading test application has been developed.

Five tests have been run, starting from 10 simulated clients and progressively increasing the number until 50 with a step of 10, using 10 different Windows-based machines (3 desktops, 5 notebooks, 3 netbooks). Each machine has been scheduled to execute a single batch script at a specified time to run all tests. The script simply calls the python application with different parameters (client to simulate, threads to use, requests to perform, time between requests), which calculates minimum, maximum and mean time spent for a single request (DataFeed modality), along with the standard deviation. As one can notice in Figure 22, the results show that the service time (time between the issued request and the reply reception) increases linearly between 10 and 40 clients, whereas it shows a non linear increment at 50 clients. However, the last case is very likely affected by a bottleneck case as 50 clients simultaneously request the DataFeed from the same set of 10 machines connected to the same network.

5. Related Work

Data stream management systems (DSMS) are among some of the most studied research subjects recently. These systems are designed to provide quick response time when dealing with large volumes of data, e.g. sensor observations. These systems employ window-based data processing combined with synopsis to process large volumes of data [10, 26, 43]. Using a synopsis helps a DSMS in reducing the response time to queries. Global Sensor Network (GSN) [2], TelegraphCQ [15], Aurora [1] and Stream [7] are among some of the known works in this domain. There are also Internet-based streaming systems, such as Stream-based Overlay Network (SBON) [48] and Peer-to-peer Information Exchange and Retrieval (PIER) [32] that process and deliver data over the Internet. They rely on P2P model for data representation, query dissemination, operations and metadata management. There exist research projects to provide access, query, streaming, and management of wireless sensor network data. The Sensor Web project [20] provides a dynamic infrastructure that allows users to access sensor networks and stream data out. Sensor Information Networking Architecture (SINA) [53] is a middleware for querying, monitoring, and tasking of sensor networks. Tiny Application Sensor Kit (TASK) [14] is built on top of TinyDB to provide high level metadata management, query configuration, monitoring and data visualization. These systems are appealing since they address the challenges related to large scale sensor resource and data sharing.

In the recent years, there have been an increasing number of initiatives to develop distributed platforms based on BSNs for e-Health applications. Many national and international research projects in academia, industry and government focus on the development and deployment of health care platforms in which wearable sensors are attached to patients for enabling round-the-clock monitoring of vital parameters. Examples of such projects include CodeBlue [41], DexterNet [38], SPINE [27, 12], SPINE2 [51], A-SPINE [3]. These systems provide abstraction from low-level TinyOS programming, but do not address the issues of integrating a Cloud infrastructure to provide extended scalability, seamless data streaming and analysis. Apart from the industry engaged in developing e-Health solution, there are also initial important investments in this field from telecommunication operators that foresee e-Health at home as a strategic business.

Only recently researchers have investigated scopes to integrate WSN with a large-scale distributed computing infrastructure. Examples include combining Cloud computing and wireless sensor network [37], Sensor grid [17] and the SensorCloud infrastructure [59]. These works describe architectural

models, case study and identify some related development issues. However, there is still a gap to develop a Cloud-based infrastructure that is targeted to BSN applications. Cloud computing technologies have also been evaluated and considered to support scientists, such as executing scientific workflow in the Cloud [19], data analysis in the Cloud [50], and high performance computing in using Amazon Web services [55]. To date, only a few research efforts have been devoted to define Cloud-enabled infrastructures seamlessly integrating BSNs and BSN data streams for supporting on-line and off-line monitoring of assisted livings.

Kahol [35] proposes a Cloud-enabled system based on an integrative gaming paradigm and designed to integrate multiple activities that involve physical exercises and cognitive skills by means of a game-based storyline. The game story performs as a motivational enabler for users to carry out multiple physical and mental activities (cycling, running, and problem solving). Such activities drive an avatar through different steps of the game. During the game phases, users have sensors on their body that can measure movements by means of accelerometers, gyroscopes, magnetometers, and gather physiological data, such as heart rate, oxygen saturation, etc. Such data, which are drivers for the game, are archived and, then, analyzed on a Cloud computing platform. The proposed Cloud-enabled system is configurable and allows researchers to easily create new games that can be driven the by different activity types. The system functionalities aim at involving and motivating users in the long term. Moreover, clinicians can exploit the system to gather clinically relevant data in a seamless way. Using the Cloud-based system, data are archived in an on-line datastore that is easily accessible by a website. This enables clinicians to remotely access the stored data and to easily integrate such data into electronic medical records. Moreover, Cloud-based tools for reporting and data analysis tools available allow for effective analysis of data and enable integration of the physiological information into biosignatures and clinical repositories. This is an important feature of the system proposed in [35] and has several implications. The final objective is that the system would provide clinicians with continuous information on their patients. However, the system proposed by Kahol [35] is special-purpose and its architecture is not reusable and extensible as basis for the development of different Cloud-enabled BSN applications.

A more general approach is proposed by Pandey et al. [46] that report the development of an autonomic Cloud environment for hosting an ECG data analysis service. In particular, they propose an autonomic Cloud environment that collects people’s health data and stores them to a Cloud-based information repository and facilitates analysis on the data using software services hosted in the Cloud. To evaluate the software design, a prototype system is developed, which is used as an experimental testbed

on a specific use case, namely, the collection of electrocardiogram (ECG) data obtained at real-time from volunteers to perform basic ECG beat analysis. The ECG software is hosted as a web-service such that any client-side implementation can simply call the underlying functions (analyze, upload data, etc.) without having to go through the complexities of the underlying application. The PaaS layer controls the execution of the software using three major components: (i) Container scaling manager, (ii) Workflow Engine, and (iii) Aneka Cloud middleware.

Although the proposed approach is more general than the previous one as it proposes a layered software architecture that could be also extended to accommodate for different BSN-oriented application services, it is still not flexible enough for rapid prototyping of community BSN applications. Moreover, it is based on Aneka that is a proprietary cloud middleware.

While such systems are special-purpose or focus on specific aspects of assisted living monitoring (physical activities, ECG, etc), our BodyCloud approach is different in that it provides real-time and off-line BSN data stream processing and analysis to support many BSN applications, using a general-purpose SaaS approach based on a widely available PaaS infrastructure, the Google App Engine.

6. Conclusions

The recent introduction of BSNs has enabled the remote monitoring of assisted livings in a broad range of application domains, such as health care, emergency management, fitness monitoring, human behavior surveillance. They can be adopted to monitor a large pool of people, thus generating large amounts of contextual data. Such a Big Data scenario requires a scalable approach for data collection, storage, processing and analysis. Cloud computing can provide a flexible storage and processing infrastructure to perform both online and offline analysis of data streams generated by BSNs.

In this paper, we have proposed BodyCloud, a Cloud-enabled SaaS architecture for the management of body sensor data streams and the complete life cycle of data analysis workflows (data collection, storing, analysis, and presentation). BodyCloud provides a platform to build and deploy applications based on community body sensor networks. System properties include rapid prototyping, scalability and flexibility of resources, ability to manage sensor heterogeneity and the dynamic deployment of user and community applications. In particular, the BodyCloud approach offers a very flexible and intuitive programming model centered on a few web-based programming abstractions (group, modality, workflow/node, view) that allow to define and deploy community BSN applications. Finally, the

definition, implementation, deployment, and testing of ECGaaS atop BodyCloud have demonstrated the aforementioned properties, confirming the effectiveness and usability of the system.

Ongoing work is currently devoted to implement a distributed rehabilitation platform based on BodyCloud and named Cloud Rehab Tutor. Specifically, such novel platform aims at supporting the rehabilitation of patients at home through BSNs able to automatically monitor the specific parameters of the body articulations (e.g. elbow, knee, ankle, wrist, shoulder) to be rehabilitated. Monitored data at the Body-side can be sent onto the Cloud-side in real-time, archived, and analyzed off-line by doctors and even by the patients themselves, and presented through specific statistics views. Moreover, doctors can upload real/virtual exercises that patients need to perform and define/modify exercise scheduling according to the rehabilitation evolution.

Future work will include:

- The definition of an intelligent and extensible component distributed between the Body-side and the Cloud-side able to support *context-aware* sensing and adaptation: contextual sensing is the ability to detect contextual information and use it to augment the user's sensory system, whereas contextual adaptation is the ability to execute or modify a service automatically based on the current context.
- The definition of a high-level global security framework for BodyCloud which allows for data privacy at the different sites (Body, Cloud, Analyst and Viewer).
- The development of other case studies which can demonstrate the range of applications which can be enabled by BodyCloud (e.g. mass fear detection system, emergency support system, etc).

7. References

[1] D. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, "Aurora: a new model and architecture for data stream management," The VLDB Journal, vol. 12, no. 2, pp. 120-139, 2003.

[2] K. Aberer, M. Hauswirth, and A. Salehi, "Infrastructure for data processing in large-scale interconnected sensor networks," Proc. Int'l Conference on Mobile Data Management (MDM'07), 2007.

[3] F. Aiello, F. Bellifemine, S. Galzarano, R. Gravina, and G. Fortino "An agent-based signal processing in-node environment for real-time human activity monitoring based on wireless body sensor networks", Journal of Engineering Applications of Artificial Intelligence, Vol. 24, n. 7, 2011, pp. 1147-1161.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," IEEE communications magazine, vol. 40, no. 8, pp. 102-114, 2002.

[5] A. Andreoli, R. Gravina, R. Giannantonio, P. Pierleoni, and G. Fortino, "SPINE-HRV: A BSN-Based Toolkit for Heart Rate Variability Analysis in the Time-Domain," in Wearable and Autonomous Biomedical Devices and Systems for Smart Environment, ser. Lecture Notes in Electrical Engineering. Springer Berlin Heidelberg, 2010, vol. 75, pp. 369–389.

[6] S. Armstrong, "Wireless connectivity for health and sports monitoring: a review," British journal of sports medicine, vol. 41, no. 5, p. 285, 2007.

[7] D. Arvind, A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom, "STREAM: The Stanford Stream Data Manager," IEEE Data Engineering Bulletin, vol. 26, 2003.

[8] A. Augimeri, G. Fortino, S. Galzarano, R. Gravina, "Collaborative Body Sensor Networks", Proc. of Int'l Conference IEEE Systems, Man and Cybertnetics (SMC2011), Oct. 9-12, Anchorage, Alaska, USA, 2011.

[9] A. Augimeri, G. Fortino, M.R. Reje, V. Handziski, A. Wolisz "A Cooperative Approach for Handshake Detection based on Body Sensor Networks," In Proc. of. IEEE International Conference on Systems, Man, and Cybernetics (SMC 2010), Istanbul, Turkey, Oct. 10-13, 2010.

[10] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," Proc. 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 1-16, ACM Press, NY, USA, 2002.

[11] L. Badger, R. Bohn, S. Chu, M. Hogan, F. Liu, V. Kaufmann, J. Mao, J. Messina, K. Mills, A. Sokol, J. Tong, F. Whiteside and D. Leaf , "US Government Cloud Computing Technology Roadmap" NIST Special Publication 500-293, Release 1.0, Volume II, 2011.

[12] F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, M. Sgroi, "SPINE: A domain-specific framework for rapid prototyping of WBSN applications" Software Practice and Experience, Wiley, 41(3), 2011, pp. 237-265, DOI:10.1002/spe.

[13] M. Berthold, N. Cebron, F. Dill, G. Di Fatta, T. Gabriel, F. Georg, T. Meinl, P. Ohl, C. Sieb, B. Wiswedel, "KNIME: the Konstanz Information Miner", Proc. of Workshop on Multi-Agent Systems and Simulation (MAS&S), 4th Annual Industrial Simulation Conference (ISC), Palermo, Italy, June 5-7, 2006, pp.58-61.

[14] P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, and S. Madden, "Task: Sensor network in a box," Proc. 2nd European Conference on Wireless Sensor Networks, pp. 133-144, 2005.

[15] S. Chandrasekaran, O. Cooper, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, F. Reiss, and M. Shah, "TelegraphCQ: Continuous dataflow processing," Proc. Int'l Conference on Innovative Data Systems Research (CIDR'03), 2003.

[16] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. C. Leung, "Body area networks: a survey," Mobile networks and applications, vol. 16, no. 2, pp. 171-193, 2011.

[17] X. Chu and R. Buyya, "Service oriented sensor web," Sensor Networks and Configuration, pp. 51-74, 2007.

[18] R. Covello, G. Fortino, R. Gravina, A. Aguilar, J.G. Breslin, "Novel method and real-time system for detecting the Cardiac Defense Response based on the ECG," IEEE Symp. MEMEA 2013, Ottawa, Canada, 2013.

[19] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," Proc. ACM/IEEE Conference on Supercomputing, p. 50, 2008.

[20] K. Delin and S. Jackson, "The sensor web: A new instrument concept," Proc. SPIE Symposium on Integrated Optics, 2001.

[21] P. Dourish, "The parting of the ways: Divergence, data management and collaborative work," Proc. 4th conference on European Conference on Computer-Supported Cooperative Work, p. 230, 1995.

[22] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," IETF RFC 2616, June 1999.

[23] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, R. Jafari, "Enabling Effective Programming and Flexible Management of Efficient Body Sensor Network Applications", IEEE Transactions on Human-Machine Systems, vol. 43, no. 1, pp. 115-133, Jan. 2013. DOI: 10.1109/TSMCC.2012.2215852.

[24] G. Fortino, M. Pathan, G. Di Fatta, “BodyCloud: Integration of Cloud Computing and Body Sensor Networks,” In Proceedings of 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2012), pp. 851-856, 2012.

[25] M. Fowler, “Patterns of Enterprise Applications Architecture,” Addison-Wesley, 2012.

[26] L. Golab and M. Özsu, "Issues in data stream management," ACM SIGMOD Record, vol. 32, no. 2, pp. 5-14, 2003.

[27] R. Gravina, A. Guerrieri, G. Fortino, F. Bellifemine, R. Giannantonio, M. Sgroi, “Development of Body Sensor Network Applications using SPINE,” In Proc. of. IEEE International Conference on Systems, Man, and Cybernetics (SMC 2008), Singapore, Oct. 12-15, 2008.

[28] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten. The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1, 2009.

[29] M. A. Hanson, H. Powell, A. T. Barth, K. Ringgenberg, B. H. Calhoun, J. H. Aylor, and J. Lach, "Body area sensor networks: Challenges and opportunities," IEEE Computer, vol. 42, no. 1, pp. 58-65, 2009.

[30] Y. Hao and R. Foster, "Wireless body sensor networks for health-monitoring applications," Physiological measurement, vol. 29, p. R27, 2008.

[31] D. Hardt, “The OAuth 2.0 Authorization Framework,” IETF RFC 6749, October 2012.

[32] R. Huebsch, B. Chun, J. Hellerstein, B. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. Yumerefendi, "The architecture of PIER: An internet-scale query processor," Proc. 2nd Biennial Conference on Innovative Data System Research, pp. 28-43, 2005.

[33] A. de Jonge, “Essential App Engine: Building High-Performance Java Apps with Google App Engine (1st ed.),” Addison-Wesley Professional, 2012.

[34] M. Jones and D. Hardt, “The OAuth 2.0 Authorization Framework: Bearer Token Usage,” IETF RFC 6750, October 2012.

[35] K. Kahol, "Integrative Gaming: A Framework for Sustainable Game-Based Diabetes Management," Journal of Diabetes Science and Technology, Volume 5, Issue 2, March 2011.

[36] D. Kirovski, N. Oliver, M. Sinclair, and D. Tan, "Health-OS: a position paper," Proc. 1st ACM SIGMOBILE international workshop on Systems and Networking Support for Healthcare and Assisted Living Environments, pp. 76-78, 2007.

[37] W. Kurschl and W. Beer, "Combining cloud computing and wireless sensor networks," Proc. 11th Int’l Conf. on Information Integration and Web-based Applications & Services, pp. 512-518, 2009.

[38] P. Kuryloski, A. Giani, R. Giannantonio, K. Gilani, R. Gravina, V. P. Seppa, E. Seto, V. Shia, C. Wang, and P. Yan, "DexterNet: An open platform for heterogeneous body sensor networks and its applications," Proc. Sixth International Workshop on Wearable and Implantable Body Sensor Networks (BSN'09), pp. 92-97, 2009.

[39] C. Leonello - www.jqplot.com

[40] B. Lo, S. Thiemjarus, R. King, and G. Z. Yang, "Body sensor network–a wireless sensor platform for pervasive healthcare monitoring," Proc. of the 3rd International Conference on Pervasive Computing (PERVASIVE 2005), vol. 13, pp. 77-80, 2005.

[41] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, "Codeblue: An ad hoc sensor network infrastructure for emergency medical care," Proc. Int’l Workshop on Wearable and Implantable Body Sensor Networks, 2004.

[42] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, “YALE: rapid prototyping for complex data mining tasks,” In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '06). ACM, New York, NY, USA, 2006, 935-940.

[43] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma, "Query processing, resource management and approximation in a data stream

management system," Proc. International Conference on Innovative Data Systems Research (CIDR'03), 2003.

[44] N. Olivr and F. Flores-Mangas, "HealthGear: automatic sleep apnea detection and monitoring with a mobile phone," Journal of Communications, vol. 2, no. 2, pp. 1-9, 2007.

[45] C. Otto, A. Milenkovic, C. Sanders, and E. Jovanov, "System architecture of a wireless body area sensor network for ubiquitous health monitoring," Journal of Mobile Multimedia, vol. 1, no. 4, pp. 307-326, 2006.

[46] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, and R. Buyya, "An Autonomic Cloud Environment for Hosting ECG Data Analysis Services," Future Generation Computer Systems, 2011.

[47] M. Patel and J. Wang, "Applications, challenges, and prospective in emerging body area networking technologies," IEEE Wireless Communications, vol. 17, no. 1, pp. 80-88, 2010.

[48] P. Pietzuch, J. Shneidman, M. Welsh, M. Seltzer, and M. Roussopoulos, "Path optimization in stream-based overlay networks," Technical Report, TR-26-04, Harvard University, 2004.

[49] C. C. Y. Poon, Y. T. Zhang, and S. D. Bao, "A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health," IEEE Comm. Magazine, vol.44, no.4, pp.73-81, 2006.

[50] X. Qiu, J. Ekanayake, S. Beason, T. Gunarathne, G. Fox, R. Barga, and D. Gannon, "Cloud technologies for bioinformatics applications," Proc. 2nd International Workshop on Many-Task Computing on Grids and Supercomputers, pp. 1-10, 2009.

[51] N. Raveendranathan, S. Galzarano, V. Loseu, R. Gravina, R. Giannantonio, M. Sgroi, R. Jafari, G. Fortino, “From Modeling to Implementation of Virtual Sensors in Body Sensor Networks”, IEEE Sensors Journal, 12(3), 2012, pp. 583-593.

[52] Sanderson, Dan (2010). Programming Google App Engine: Build and Run Scalable Web Apps on Google's Infrastructure. O'Reilly Media. ISBN 978-0-596-52272-8.

[53] C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor information networking architecture and applications," IEEE Wireless Communications, vol. 8, no. 4, pp. 52-59, 2001.

[54] SpryMedia © 2008-2013 - www.datatables.net

[55] J. Varia, "Architecting Applications for the Amazon Cloud," in Cloud Comptuing: principles and Paradigms, R. Buyya, et al., Eds., ed: Wiley, NY, USA, 2011.

[56] G.S. Wagner, Marriott’s Practical Electrocardiography, Lippincott Williams & Wilkins, Ed., 2007.

[57] G. Z. Yang, Body sensor networks: Springer-Verlag, 2006.

[58] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292-2330, 2008.

[59] M. Yuriyama and T. Kushida, "Sensor-Cloud Infrastructure-Physical Sensor Management with Virtualized Sensors on Cloud Computing," *Proc. International Conference on Network-based Information Systems (NBiS'10)*, pp. 1-8, 2010.